

(12) UK Patent Application (19) GB (11) 2 345 620 (13) A

(43) Date of A Publication 12.07.2000

(21) Application No 9925678.6

(22) Date of Filing 29.10.1999

(30) Priority Data

(31) 60106374 (32) 30.10.1998 (33) US
(31) 09404955 (32) 24.09.1999

(71) Applicant(s)

Citrix Systems Inc
(Incorporated in USA - Florida)
6400 NW 6th Way, Fort Lauderdale, Florida 33309,
United States of America

(72) Inventor(s)

John Albert Bull
David John Otway

(74) Agent and/or Address for Service

Frank B Dehn & Co
179 Queen Victoria Street, LONDON, EC4V 4EL,
United Kingdom

(51) INT CL⁷

H04L 9/08

(52) UK CL (Edition R)

H4P PDCSP
U1S S2124 S2209

(56) Documents Cited

EP 0851628 A1 US 5491750 A

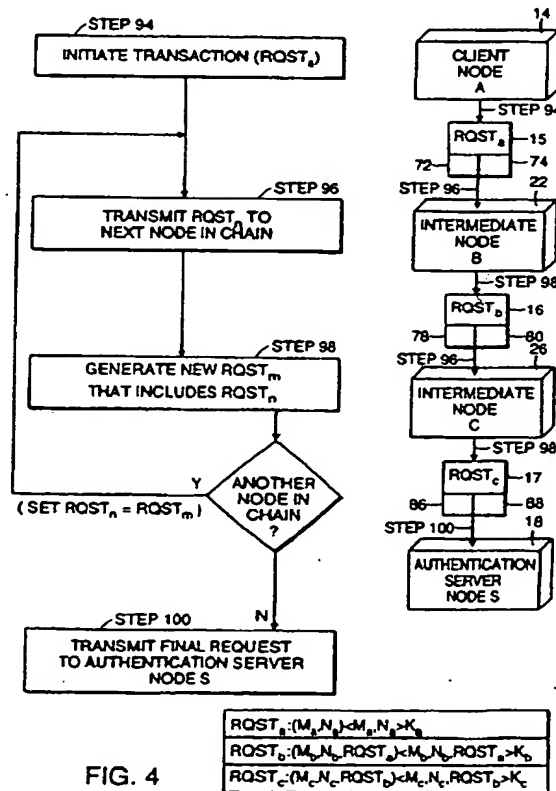
(58) Field of Search

UK CL (Edition R) H4P PDCSP PPEB
INT CL⁷ H04L 9/08 12/22
Online: EPODOC, JAPIO, WPI

(54) Abstract Title

Secure distribution of session keys to a chain of network nodes

(57) A chain of nodes 14, 22, 26 recursively constructs steps 94, 96, 98 and presents step 100 a nested request to the authentication server 18, which includes a sealed or encrypted portion. The nested request includes a request from each of the nodes in the chain requiring a session key to communicate with a neighboring node. The authentication server recursively unravels the request and recursively prepares a response that includes a session key for each node that submitted a request Figs. 5, 6, 7. The response traverses the chain of nodes in the reverse order taken by the nested request to reach the authentication server. Each node receiving the response extracts the portion of the response directed to that node, and forwards the remainder of the response, if any, to the next node in the chain Fig 8. Thus, with a single traversal of the chain of nodes each node receives at least one session key. The forward and reverse protocols easily generalize for any number of nodes in the chain. The protocols can employ one-way hash functions to seal requests and responses and to encode/encipher session keys. Portions of the response may be encrypted using the session keys.



GB 2 345 620 A

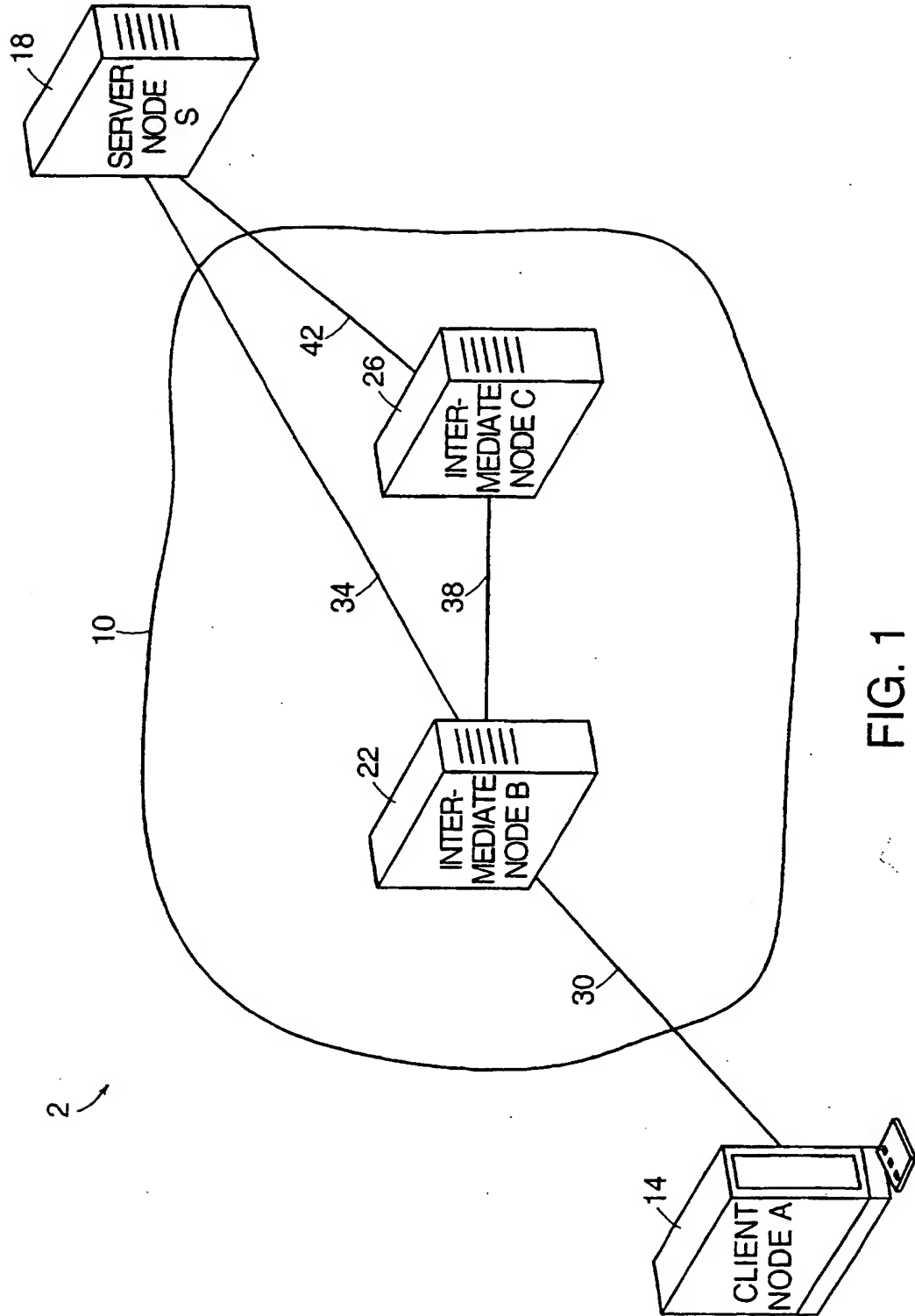


FIG. 1

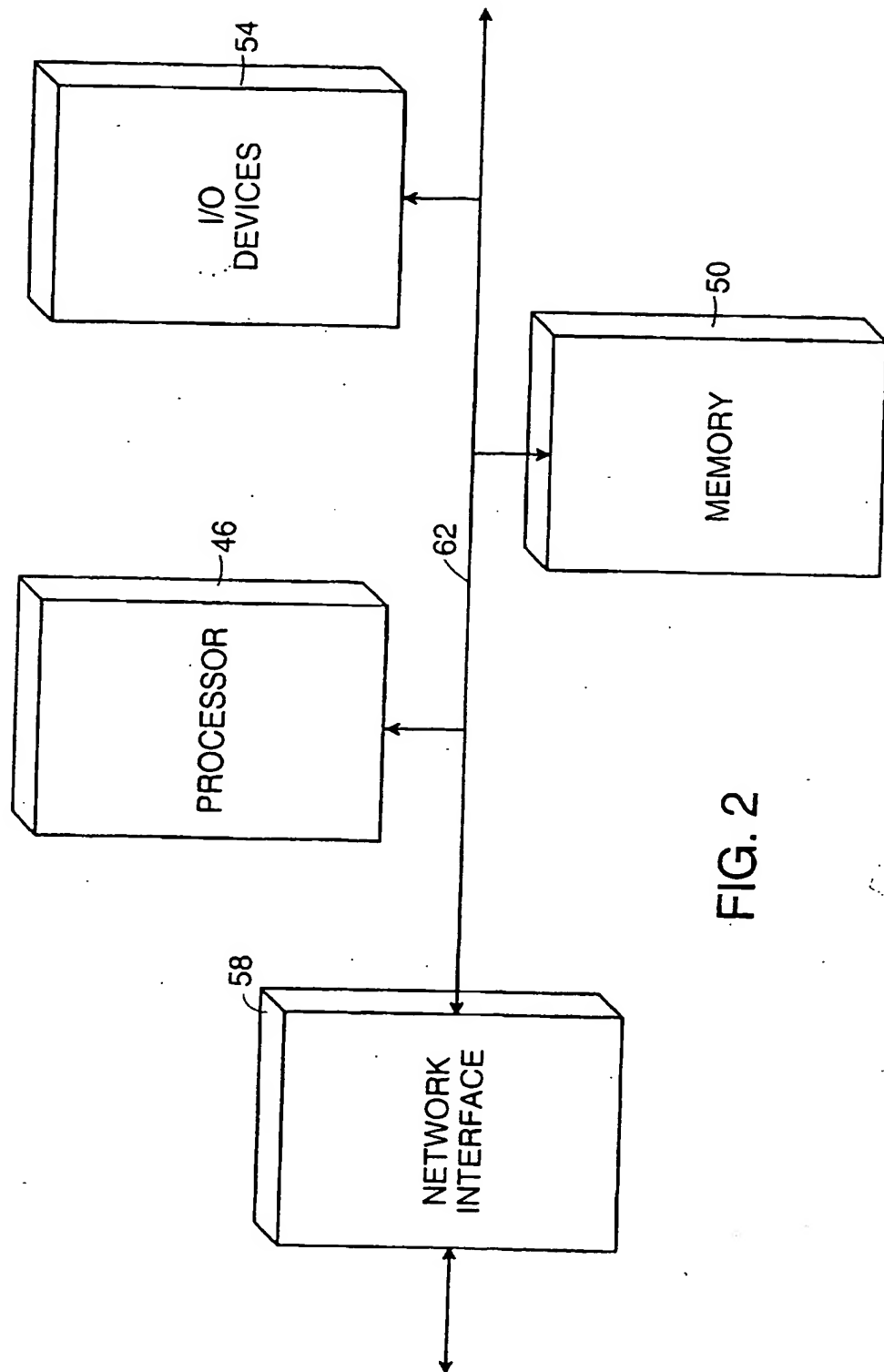


FIG. 2

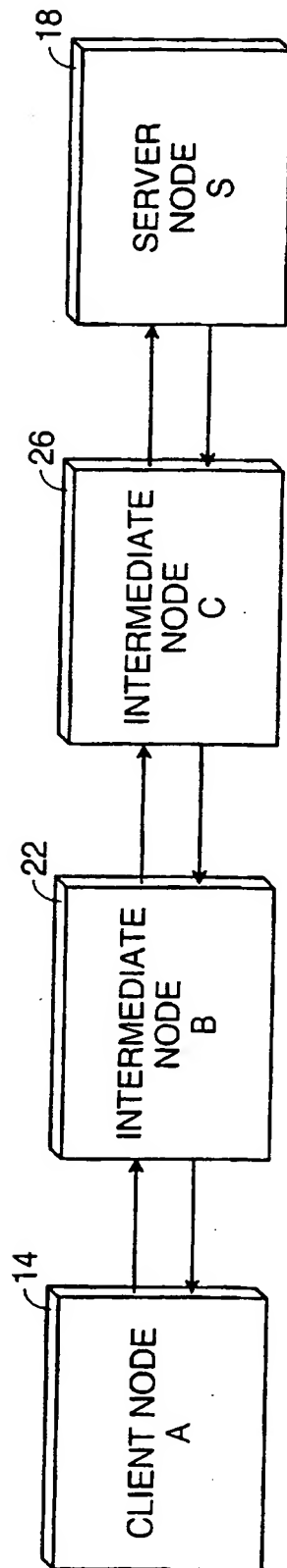


FIG. 3

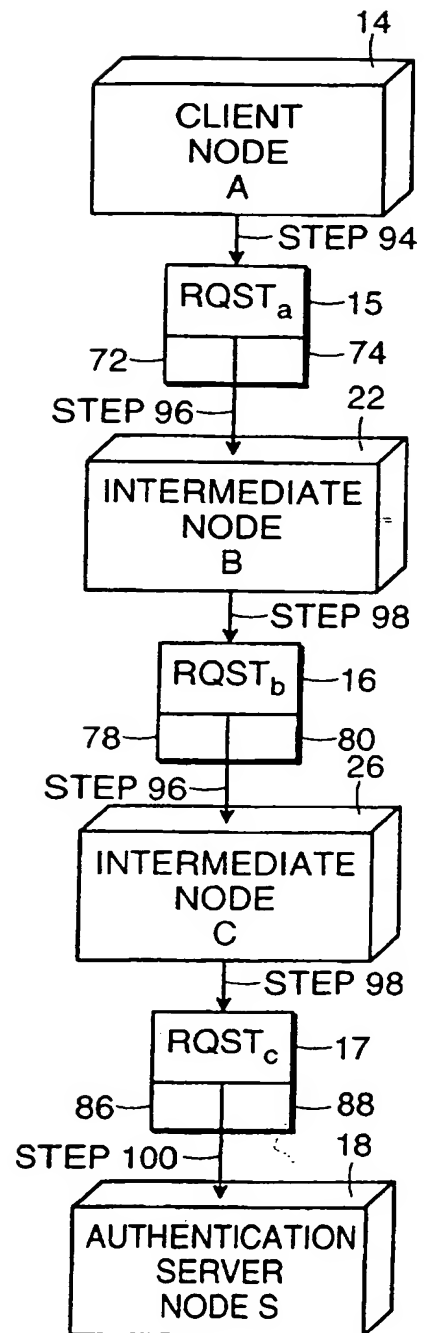
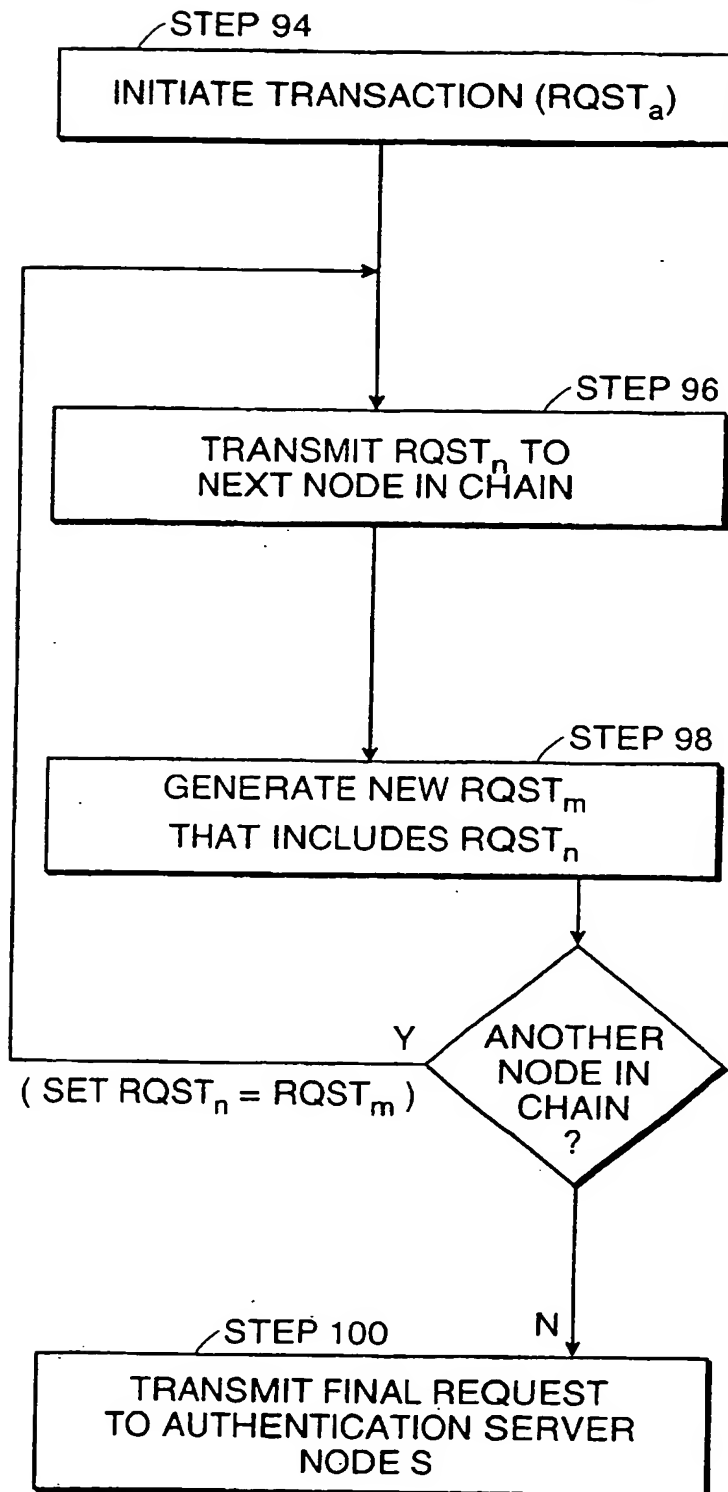


FIG. 4

$$RQST_a: (M_a, N_a) < M_a, N_a > K_a$$

$$RQST_b: (M_b, N_b, RQST_a) < M_b, N_b, RQST_a > K_b$$

$$RQST_c: (M_c, N_c, RQST_b) < M_c, N_c, RQST_b > K_c$$

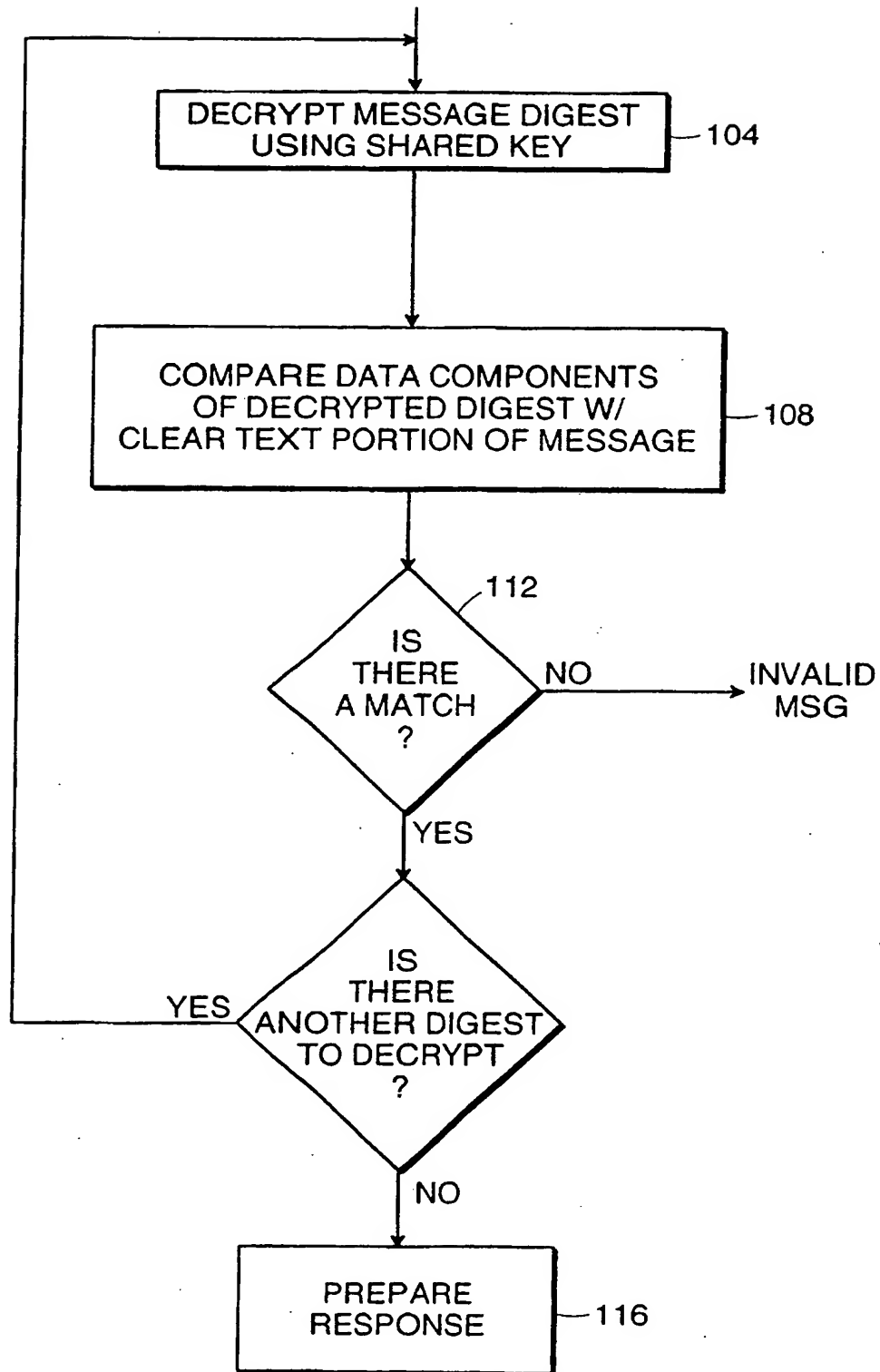
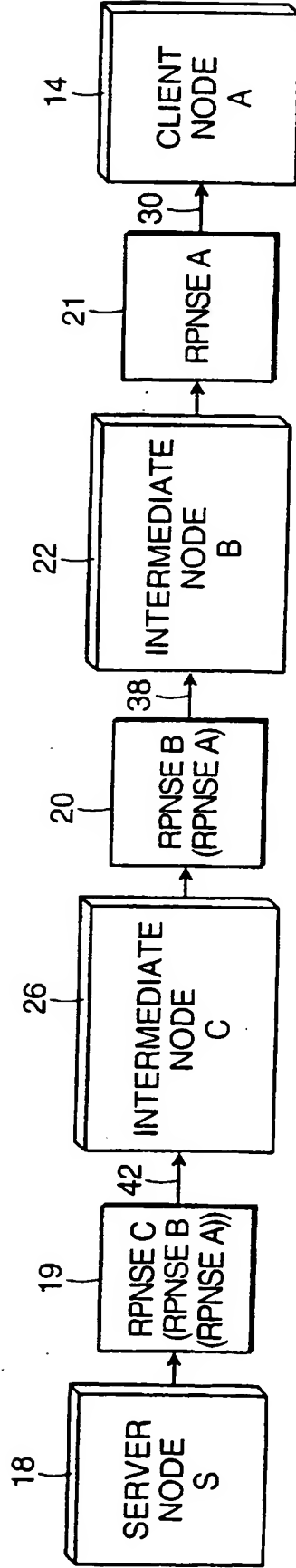


FIG. 5



6/9

RPNSE C: $(R_{cb}, N_c, N_b, \text{en}K_{cb}, \text{RPNSE B}) < R_{cb}, N_c, N_b, \text{en}K_{cb}, \text{RPNSE B} > K_{cb}$
RPNSE B: $(R_{bc}, N_b, N_c, \text{en}K_{bc}, R_{ba}, N_b, N_a, \text{en}K_{ba}, \text{RPNSE A} < R_{ba}, N_b, N_a, \text{en}K_{ba}, \text{RPNSE A} > K_{ba}),$ $< R_{bc}, N_b, N_c, \text{en}K_{bc}, R_{ba}, N_b, N_a, \text{en}K_{ba}, \text{RPNSE A} > R_{ba}, N_b, N_a, \text{en}K_{ba}, \text{RPNSE A} > K_{ba} > K_{bc}$
RPNSE A: $(R_{ab}, N_a, N_b, \text{en}K_{ab}) < R_{ab}, N_a, N_b, \text{en}K_{ab} > K_{ab}$

FIG. 6

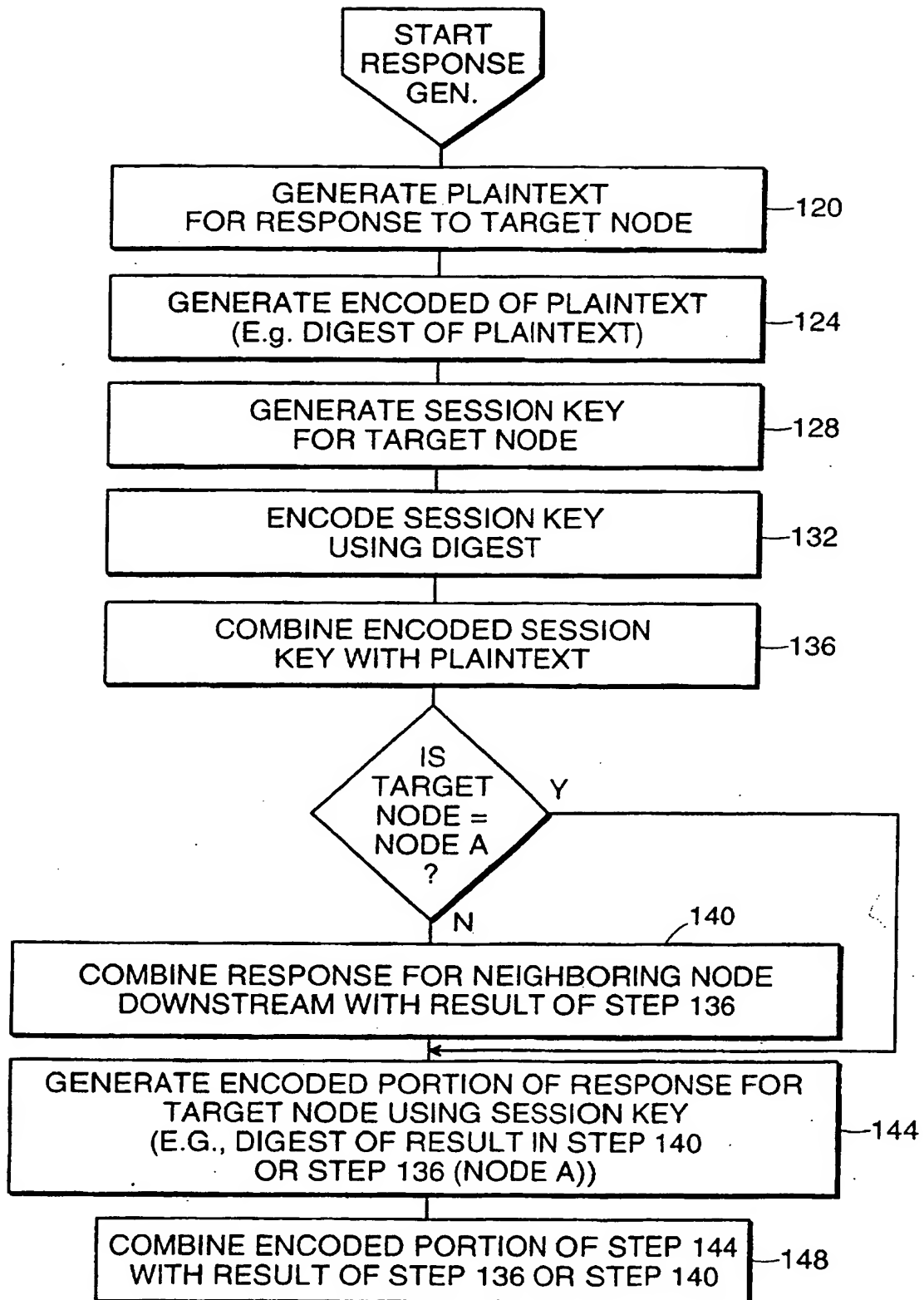


FIG. 7A

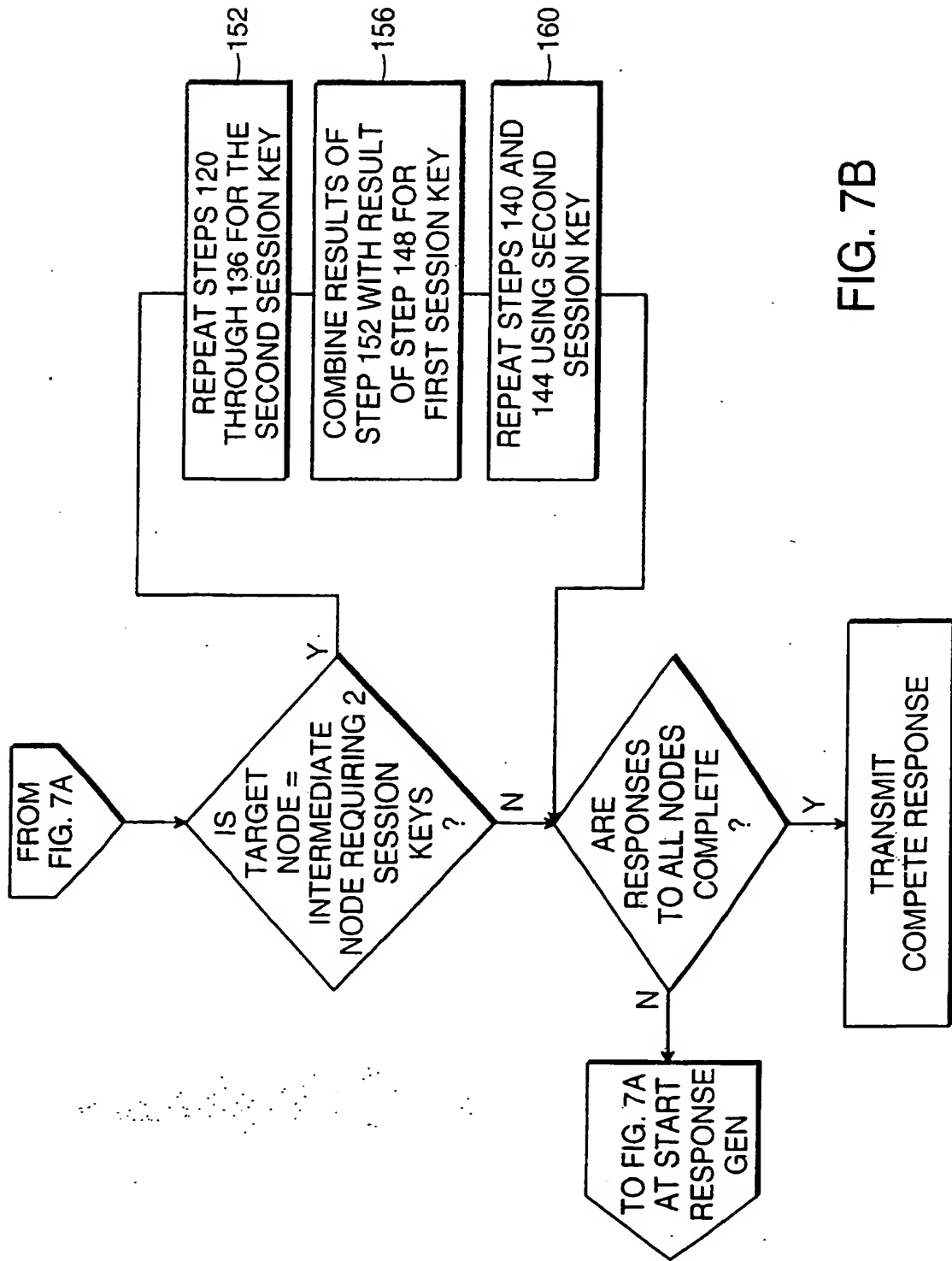


FIG. 7B

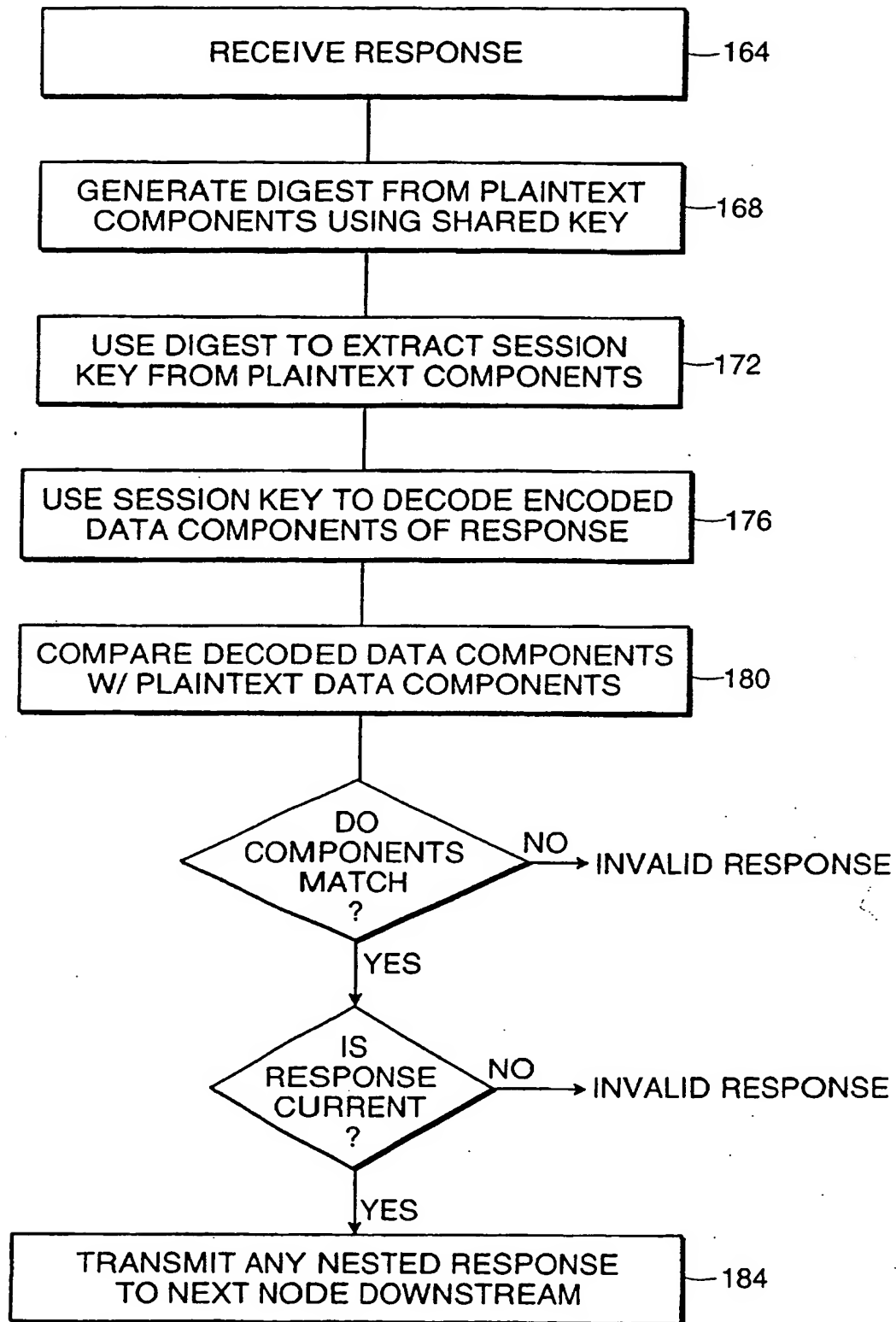


FIG. 8

A SYSTEM AND METHOD FOR SECURE DISTRIBUTION OF SESSION KEYS TO A CHAIN OF COMPUTER SYSTEM NODES IN A NETWORK

This invention relates generally to client-server computer networks. More specifically, the invention relates to a system and method for securely distributing information among clients and servers in a network.

Authentication of computer systems plays an important role in data communications over modern networks. With the rapidly increasing reliance on the electronic highways to convey sensitive data comes the greater need for increased security for such data transmissions. Computer systems need to be mutually assured of the identities of those computer systems with which they exchange information. Further, these computer systems need the assurance that the information in these communications has not been altered during transmission. These needs have led to various techniques that enable computer systems to exchange information securely.

One common authentication technique entails presenting a challenge to the computer system to which the computer system must correctly respond in order to gain permission for subsequent communication. Other authentication techniques involve encryption methods. Generally, there are two main types of encryption methods: asymmetric encryption and symmetric encryption. Asymmetric encryption methods use two different keys, one to encrypt the communication and the other to decrypt the communication. For example, public-key encryption is an asymmetric encryption technique in which one computer system encrypts a communication using a public key and another computer system decrypts the

communication using a private key known only to that other computer system. In contrast, symmetric encryption uses one key for both encryption and decryption. Some authentication techniques combine asymmetric and symmetric encryption methods. One exemplary technique is to use public key encryption to distribute a session key to a pair of computer systems that these computer systems then use with symmetric encryption algorithms to exchange encrypted data communications.

An important factor to be considered when using encryption algorithms, however, is that some countries limit the key size for encryption within exported computer and software products. It is understood by those skilled in the art that such encryption algorithms, when constrained by the key size, may be broken.

Summary of the Invention

According to one aspect of the present invention, in a network including a first node, a second node, and a third node, there is provided a method for securely delivering digital information to the first node from the third node by way of the second node, the method comprising the steps of:

- receiving a request at the third node from the first node;
- generating digital information in response to the request;
- operating on the request and the digital information to produce a first data structure, the first data structure including a representation of the digital information;
- operating on the request and the first data structure to produce a second data structure, the second data structure including the first data structure; and
- transmitting the second data structure to the second node.

According to another aspect of the invention, in a network including a client node and an authentication server node, there is provided a method for securely delivering a session key to the client node from the authentication server node in response to a request from the client node, the method comprising the steps of:

sealing plaintext using the session key;

encoding the session key using a key shared with the client node; and

transmitting a data structure to the client node that includes the encoded session key and the sealed plaintext.

According to another aspect of the invention there is provided a system for securely distributing a session key by way of a network, the network including a first node transmitting a request to obtain the session key and a second node in communication with the first node, the system comprising:

a third node in communication with the second node and receiving the request by way of the second node, the third node including:

a processor generating (1) a first data structure by operating on the request and the session key, the first data structure including a representation of the session key, and (2) generating a second data structure by operating on the request and the first data structure, the second data structure including the first data structure; and

a network interface coupled to the processor for transmitting the second data structure to the second node over the network.

In one embodiment, the invention features a method for securely delivering digital information to the first node from the third node by way of the second node. The method includes receiving a request at the third node from the first node. In response to the request,

digital information is generated. The request and the digital information are then operated on to produce a first data structure. The first data structure includes a representation of the digital information. The request and the first data structure are then operated on to produce a second data structure, with the second data structure including the first data structure. The second data structure is transmitted to the second node.

In one embodiment, the digital information includes a session key for the first node to use when communicating with the second node. The session key is encoded using a key shared exclusively with the first node to conceal the session key within the first data structure. Also, the session key can be used to seal a portion of the first data structure. A second session key can be generated for the second node to use in communications with the first node. This second session key can be used to seal a portion of the second data structure containing the first data structure. Also, the second session key can be encoded using a key shared exclusively with the second node. The second data structure includes the encoded second session key.

In one embodiment, the invention features a method for securely delivering a session key to a client node from an authentication server node in response to a request from the client node. The method includes sealing plaintext using the session key. The session key is encoded using a key shared with the client node. A data structure including the encoded session key and the sealed plaintext is transmitted to the client node. At the client node, the data structure can be extracted. The encoded session key is decoded using the shared key, and the seal of the plaintext checked using the decoded session key. The plaintext can be used to authenticate that the session key originated from the authentication server, that the decoded session key is unaltered during transmission from the authentication server, and that

the data structure is a current response from the authentication server to the request from the client node.

In one embodiment, the invention features a system for securely distributing a session key by way of a network. The network includes a first node transmitting a request to obtain the session key and a second node in communication with the first node. The system also includes a third node in communication with the second node for receiving the request by way of the second node. The third node has a processor that generates a first data structure by operating on the request and the session key. The resulting first data structure includes a representation of the session key. The processor also generates a second data structure by operating on the request and the first data structure. The second data structure includes the first data structure.

Some embodiments of the invention will now be described in more detail with reference to the accompanying drawings, in which:

Fig. 1 is a diagram of an embodiment of a client node in communication with an authentication server node over a network;

Fig. 2 is a diagram of an embodiment of functional components within each computer node;

Fig. 3 is a diagram showing two exemplary forward flows of requests from the client node through two intermediate nodes to the authentication server node;

Fig. 4 is a flow chart and block diagram representation of an embodiment of a process by which embedded requests are generated;

Fig. 5 is a flow chart representation of an embodiment of a process by which the authentication server processes the embedded requests;

Fig. 6 is a diagram showing an exemplary return flow of responses from the authentication server node to the client node by way of the first and second intermediate nodes;

Figs. 7A and 7B provide a flow chart representation of an embodiment of a process by which the authentication server generates a response having at least one session key for each node in the chain of nodes; and

Fig. 8 is a flow chart representation of an embodiment of a process by which each node receiving a response extracts at least one session key from that response.

Fig. 1 shows an exemplary client-server system 2 embodying the principles of the invention. The client-server system 2 includes a computer system (client node A) 14 in communication with another computer system (authentication server node S) 18 over a network 10 through a first and a second intermediate computer systems (intermediate nodes B and C) 22 and 26, respectively. It is to be understood that the system 2 can include any number of additional client, intermediate, and server nodes. Some of such intermediate nodes can merely forward information without participating in the authentication process. The network 10 can be, for example, a wide area network (WAN) or a local area network (LAN), an international network of networks, such as the Internet and the World Wide Web, or a network within a single organizational entity, called an Intranet.

Fig. 2 shows functional components of an exemplary embodiment of the each node 14, 18, 22, and 26. Each node 14, 18, 22, and 26 can be any conventional personal computer, workstation, network terminal, mini-computer, or mainframe computer. Each can include a processor 46, memory 50 for storing data and software programs, and I/O devices 54 (e.g., a keyboard and/or a mouse) in communication by way of a signal bus 62. Each node 14, 18,

22, 26 also includes a network interface 58 for communicating with the network 10. Stored in the memory 50 of each node are software routines for performing the authentication protocol of the invention as described below. These software routines (1) generate authentication requests, (2) evaluate authentication requests, (3) generate responses to authentication requests, and (4) evaluate responses. The processor 46 of each node executes the appropriate software routines to effectuate a forward flow (i.e., toward the server S 18) of authentication requests and a return flow (i.e., away from the server S 18) of responses as described below.

Referring again to Fig. 1, two exemplary communication paths traverse the network 10 from the client node A 14 to the authentication server node S 18. Each communication path can be represented as a "chain of nodes." A first chain of nodes includes the client node A 14, the first intermediate node B 22, and the authentication server node S 18. The first intermediate node B 22 is in communication with the client node A 14 by way of the communication link 30 and with the authentication server node S 18 by way of the communication link 34.

A second exemplary chain of nodes includes the client node A 14, the first intermediate node B 22, the second intermediate node C 26, and the authentication server node S 18. The first intermediate node B 22 is in communication with the client node A 14 by way of communication link 30 and with the second intermediate node C 26 by way of communication link 38. The second intermediate node C 26 is also in communication with the authentication server node S 18 by way of a communication link 42. The two chains of nodes are merely exemplary; a chain of nodes can have any number of intermediate nodes between the client node A 14 and the authentication server node S 18 that employ the

authentication protocol. Other embodiments include intermediate nodes that may not seek authentication or session keys, and therefore do not use the authentication protocol, but rather operate in the chain of nodes to forward requests and responses to the next neighboring node in the chain.

The client node A 14 initiates authentication requests. For example, a user of the client node A 14 can submit a digital signature to approve a transaction. The first intermediate node B 22 can be a client-server node. Examples of intermediate nodes include a counter signer for approving financial transactions exceeding a dollar amount limit, a corporate signer for validating the signature of the client node A 14, a sacrificial server disposed between the two firewalls, and a business process server in front of a back end database server.

Each node 14, 18, 22, 26 can function in the network 10 in the capacity of a client node, an intermediate node, or an authentication server depending upon the position of that node in a chain of nodes. For example, the client node A 14 and the authentication server node S 18 can function as an intermediate node when that node 14 or 18 receives a request to be forwarded to another authentication server. Each intermediate node 22, 26 and the authentication server node S 18 can operate as a client intermediate node 22 by initiating an authentication request. The client node A 14 and each intermediate node 22, 26 can function in the role of an authentication server.

The authentication protocol assumes that each node 14, 22, and 26 shares a secret key exclusively with the authentication server 18. In the following description, the key shared by the client node A 14 and the authentication server node S 18 is referred to as K_a , by the first intermediate node B 22 and the authentication server node S 18 is K_b , and by the second

intermediate node C 26 and the authentication server node S 18 is K_c . The client node A 14 and each intermediate node 22, 26 trusts the authentication server node S 18. That is, each node 14, 22, 26 accepts that the authentication server 18 correctly performs the function of identifying the nodes 14, 22, 26 by way of the secret keys that the server node S 18 shares with each node 14, 22, 26

In the example shown, the nodes 14, 18, 22, 26 communicate using an authentication protocol in accordance with the principles of the invention. When used between nodes needing authentication, the authentication protocol operates to provide authentication between pairs of nodes in a chain of nodes in the client-server system 2. Authentication occurs at various points in the protocol. First, each node receiving a message is assured of the identity of the node transmitting the message so that communications are not conducted with an impostor. Second, the protocol assures that the contents of the received message are authentic in that such contents have not be altered during transmission from the transmitting node to the receiving node. Third, the protocol assures the timeliness of the received message in that the message is received in response to a particular request and is not an attempt by an intruder to mimic a previous authentic communication.

The authentication protocol also produces session keys for each pair of nodes as described below. Each session key can be a randomly generated value used for signing messages or encryption. Only nodes possessing the same session key for decoding the encryption can understand encrypted communications. Consequently, two or more nodes having the same session key can communicate privately. Session keys exist only for the duration of a particular session (or conversation). Accordingly, nodes can request session

keys as needed and subsequently destroy the session keys when the conversations are complete.

Fig. 3 shows an exemplary chain of nodes illustrating a bi-directional flow of requests and responses. Requests pass to and responses pass from the authentication server node S 18. In brief overview, in one embodiment the client node A 14 generates a request for authentication and for obtaining a session key and transmits the request to the first intermediate node B 22. The request includes both plain text and sealed text. The first intermediate node B 22 adds its authentication information to the original request from the client node A 14 to generate a new authentication request and forwards the new authentication request to the second intermediate node C 26. Nested within the new authentication request is the original request from the client node A 14.

The second intermediate node C 26 adds information to the request from the first intermediate node B 22 to generate a second new authentication request and forwards the second new authentication request to the authentication server node S 18. Nested within the second new authentication request is the authentication request received from the first intermediate node B 22, within which is the original request from the client node A 14. The process can be repeated for as many intermediate nodes as are in the chain of nodes. The authentication server node S 18 receives a final request containing the nested requests of each intermediate node 22, 26 and the client node A 14.

According to the principles of the invention, the authentication server node S 18 recursively unravels the final request containing the nested requests to determine those nodes in the chain that are expecting a response. From the unraveling of the final request, the authentication server node S 18 determines an order in which to recursively nest responses to

each of the nodes in the chain within a final response. Consequently, the final response includes a response to each node in the chain, and each response includes at least one session key for that respective node in the chain. The individual responses are nested within each other, as described further below, with the response to the client node A 14 being most deeply nested within the final response.

In the embodiment shown, the authentication server node S 18 transmits the final response to the second intermediate node C 26. Each node in the chain extracts and authenticates at least one session key directed to that node from the final response, and forwards the remainder of the response to the next node in the chain away from the server 18. Accordingly, with the final response the authentication server node S 18 delivers session keys and provides authentication to each node in a chain. It is to be understood that the principles of the invention can be extended to securely transmit any type of information and not just session keys. Also, the protocol can extend to any number of nodes in a chain. An advantage gained by the authentication protocol of the invention is that each node in the chain does not need knowledge of the global behavior of the system 2 to achieve authentication and obtain a session key.

In one embodiment, the authentication server node S 18 can deliver the same session key to pairs of adjacent nodes, such as, for example, the client node A 14 and the first intermediate node B 22. In other embodiments, the authentication server node S 18 can deliver the same session key to nodes that are separated by one or more intermediate nodes, e.g., the client node A 14 and the second intermediate node C 26.

Generation of an Authentication Request

Fig. 4 presents a flowchart and a block diagram illustrating an embodiment of an exemplary process by which the client node A 14 and each intermediate node 22, 26 generate and forward a recursively constructed final request to the authentication server node S 18. The client node A 14 initiates (step 94) an authentication request RQSTa 15. In the embodiment shown, RQSTa 15 includes a plaintext portion 72 and a sealed version 74 of the plaintext portion 72. The sealed version 74 is sealed by the client node A 14 using the shared key K_a as described previously and below. Throughout the specification, the sealing of data (here, a copy of the plaintext portion 72) is accomplished by applying a one-way hash-function to the data. One-way hash functions take variable-length input strings and produce fixed-length output strings. An attribute of such a one-way hash function is that reconstructing the input string from the output string is computationally difficult. Examples of one-way hash functions that can be used to practice the invention are well known in the art.

A purpose of the one-way hash function is to guarantee the integrity of the message data. For example, when the server S 18 receives the plaintext portion 72 and the sealed version 74 from the client node A 14, the server S 18 can apply the one-way hash function to the plaintext portion 72 using the secret key K_a shared with the client node A 14. Thus, the server S 18 produces another sealed version of the plaintext portion 72, which the server S 18 can compare with the sealed version 74 received from the client node A 14. A match between the sealed versions indicates that the integrity of the data was maintained during the transmission from the client node A 14 to the server node S 18. Sealing can be accomplished by employing other techniques (e.g., encryption algorithms). Accordingly, any reference

hereafter to sealing or sealed data contemplates any method, e.g., a one-way hash function or an encryption algorithm, capable of producing the seal.

The plaintext portion 72 is a bit string of data that includes a nonce Na combined with a message Ma . The nonce Na is a value randomly generated by the client node A 14 that uniquely identifies the authentication request at client node A 14. Client node A 14 stores the nonce Na in memory for later reference. The message Ma is a bit string that identifies the two nodes for which a session key is sought, here client node A 14 and the intermediate node B 22. An exemplary notation for the message Ma is (A, B). In one embodiment, the nonce Na is concatenated to the end of the message Ma such that an exemplary notation for the plaintext portion 72 is

(Ma, Na) .

The sealed version 74 is a sealed bit string produced from the data Ma, Na in the plaintext portion 72 and is referred to as message digest, Da . The client node A 14 can produce the message digest Da by computing a one-way hash function of the plaintext data Na and Ma using the key, Ka , shared exclusively with the authentication server node S 18. Before generating the message digest Da , the client node A 14 inserts a copy of the key Ka into a copy of the plaintext portion 72. The key Ka can be placed in front of the copy of the plaintext portion 72 to add randomness to the message and/or at the end of the copy of the plaintext portion 72 to prevent the digest Da from being extended. Although the invention does not require placement at both the front and at the end, performance of both strengthens the protection of the message digest Da . An exemplary notation for the message digest Da is:

$\langle Ma, Na \rangle Ka$.

To form RQST_a 15, the plaintext portion 72 and sealed version 74 are combined. In one embodiment, the sealed version 74 is concatenated to the end of the plaintext portion 72.

An exemplary notation for the RQST_a 15 is:

$(Ma, Na), Da$ or

$(Ma, Na), <Ma, Na>Ka$.

Note that the parenthesis “(),” braces “< >,” spaces and commas are for notation purposes only and are not part of the string of data that form RQST_a 15 (or any other bit string of data).

The client node A 14 transmits (step 96) the RQST_a 15 to the intermediate node B 22 by way of communication link 30. Responsive to the RQST_a 15, the intermediate node B 22 produces (step 98) a new request RQST_b 16. Again, the RQST_b 16 includes a plaintext portion 78 and a sealed version 80 of the plaintext portion 78 as described previously with respect to RQST_a 15. The plaintext and sealed portions 78, 80 each include the plaintext portion 72 and the sealed version 74 of RQST_a 15.

Specifically, the plaintext portion 78 is a bit string that includes a message, Mb , and a nonce Nb generated by intermediate node B 22, and the RQST_a 15 is received from client node A 14. Intermediate node B 22 stores the nonce Nb in memory for later reference when verifying a response from server node S 18. An exemplary notation for the plaintext portion 78 is

$(Mb, Nb), (Ma, Na), <Ma, Na> Ka$ or

$(Mb, Nb, RQST_a)$.

Note that the plaintext portion 78 includes sealed data, which come from RQST_a 15.

In this embodiment, intermediate node B 22 concatenates the nonce Nb to the message Mb , and $RQSTa$ 15 to the end of the nonce Nb . In other embodiments, the order may differ. As with message Ma , the message Mb includes the identities of the nodes for which a session key is requested. Thus, if the session key is required between intermediate node B 22 and intermediate node C 26, the message Mb is (B, C).

Similar to previous $RQSTa$ 15, the sealed version 80 is an sealed bit string representing the data in the plaintext portion 78. In one embodiment, the sealed version 80 includes a message digest, Db , produced by computing a one-way hash function of the data components Mb , Nb , and $RQSTa$ 15 using the key, Kb , shared exclusively by nodes B and S. An exemplary notation for the sealed version 80, which is the message digest Db , is

$\langle Mb, Nb, Ma, Na, \langle Ma, Na \rangle Ka \rangle Kb$ or

$\langle Mb, Nb, RQSTa \rangle Kb$.

In step 98, the intermediate node B 22 generates the new request $RQSTb$ 16 as a combination of the plaintext portion 78 and the sealed version 80. In one embodiment, the sealed version 80 is concatenated to the end of the plaintext portion 78 as described previously and therefore an exemplary notation for the $RQSTb$ 16 is

$(Mb, Nb, RQSTa), \langle Mb, Nb, RQSTa \rangle Kb$,

in which the $RQSTa$ 15 is recursively nested within the $RQSTb$ 16. Expanded notation for $RQSTb$ is:

$(Mb, Nb, (Ma, Na), \langle Ma, Na \rangle Ka), \langle Mb, Nb, (Ma, Na), \langle Ma, Na \rangle Ka \rangle Kb$.

Similarly as for client node A 14 and intermediate node B 22, intermediate node C 26 produces a new request $RQSTc$ 17. $RQSTc$ 17 includes a plaintext portion 86 and a sealed version 88 of the plaintext portion 86.

The plaintext portion 86 is a bit string that includes a message M_c , a nonce N_c , and the $RQST_b$ 16. Exemplary notation here for M_c is (C, S) , and for the plaintext portion 86, $(M_c, N_c, RQST_b)$. Intermediate node C 26 stores the nonce N_c in memory for use when verifying a response from server node S 18. Again, the message M_c includes the identities of the two intermediate nodes for which a session key is sought.

The sealed version 88 is a sealed bit string representing the data, M_c , N_c , and $RQST_b$ 16, in the plaintext portion 86. In one embodiment, sealed version 88 includes a message digest, D_c , produced by computing a one-way hash function of the data components M_c , N_c , and $RQST_b$ 16 using the key K_c . K_c is the key shared exclusively between intermediate node C 26 and server node S 18. An exemplary notation for the message digest D_c is

$$\langle M_c, N_c, RQST_b \rangle K_c.$$

In step 98, the intermediate node C 26 generates the new request $RQST_c$ 17 as a combination of the plaintext portion 86 and the sealed version 88, for example, by concatenating the sealed version 88 to the end of the plaintext portion 86. An exemplary notation for $RQST_c$ 17 is

$$(M_c, N_c, RQST_b), \langle M_c, N_c, RQST_b \rangle K_c.$$

in which the $RQST_b$ 16 is recursively nested within the $RQST_c$ 17. Expanded notation for $RQST_c$ is:

$$(M_c, N_c, (M_b, N_b, (M_a, N_a), \langle M_a, N_a \rangle K_a), \langle M_b, N_b, (M_a, N_a), \langle M_a, N_a \rangle K_a \rangle K_b) \\ \langle M_c, N_c, (M_b, N_b, (M_a, N_a), \langle M_a, N_a \rangle K_a), \langle M_b, N_b, (M_a, N_a), \langle M_a, N_a \rangle K_a \rangle K_b \rangle K_c.$$

The form of the request generated at each node X requesting a session key for communication with node X+1 generalizes to:

$$RQST_x = (X, X+1, N_x, RQST_{x-1}), \langle X, X+1, N_x, RQST_{x-1} \rangle K_x,$$

where X-1 is the identity of the previous node, if any, and RQST_{x-1} is the request given by the previous node X-1 for requesting a session key for communication between node X-1 and X.

Evaluating the Authentication Request

In step 100, the last intermediate node in the chain of nodes transmits the final request to server node S 18. For illustration purposes, the last intermediate node in the example shown is intermediate node C 26, and the final request is RQST_c 17. Accordingly, as described above, the final request RQST_c 17 received by server node S 18 can be expressed as:

$$(Mc, Nc, RQSTb) <Mc, Nc, RQSTb>Kc,$$

where the plaintext portion 86 is

$$(Mc, Nc, RQSTb),$$

and the sealed version 88 of the plaintext portion 86 is

$$<Mc, Nc, RQSTb>Kc.$$

Fig. 5 shows an exemplary process by which the server node S 18 recursively unravels RQST_c 17 to (1) authenticate each node in the chain of nodes and (2) authenticate the integrity of the data in each request within RQST_c 17 (i.e., RQST_b 16 and RQST_a 15). The server node S 18 uses (step 104) the key shared with intermediate node C 26, K_c, to determine the message digest D_c. From the digest D_c, the server node S 18 extracts the data Mc, Nc, and RQST_b 17 from the sealed version 88.

In step 108, server node S 18 compares the extracted data, Mc, Nc, and RQST_b 16, with the data Mc, Nc, and RQST_b 16 in the plaintext portion 86. If the extracted data match the plaintext data, the server node S 18 knows that intermediate node C 26 is the sender of the RQST_c 17 because, other than server node S 18, only intermediate node C 26 knows the

shared key K_c used to create the sealed version 88. Server node S 18 therefore also knows that the data M_c , N_c , and $RQST_b$ 16 are unaltered during the transmission from intermediate node C 26 because the extracted and the plaintext data are the same. If the extracted data do not match the plaintext data, the final request $RQST_c$ 17 is invalid. The server node S 18 can then ignore such a request.

If $RQST_c$ 17 is determined to be valid, server node S 18 then evaluates $RQST_b$ 16, which can be expressed as:

$$(M_b, N_b, RQST_a), \langle M_b, N_b, RQST_a \rangle K_b.$$

The plaintext portion 78 is therefore

$$(M_b, N_b, RQST_a)$$

and the sealed version 80 of the plaintext portion 78 is

D_b or

$$\langle M_b, N_b, RQST_a \rangle K_b.$$

Server node S 18 then uses K_b to extract (step 104) the data M_b , N_b , and $RQST_a$ 15 from the sealed version 80 and again compares (step 108) the extracted data with the data M_b , N_b , and $RQST_a$ 15 of the plaintext portion 78.

In step 112, the extracted data are compared with the plaintext data components and if the components match, intermediate node B 22 is the sender of the $RQST_b$ 16 because only nodes B and S know the shared key K_b used to create the sealed version 80. Again, this match shows that the data M_b , N_b , and $RQST_a$ 15 are unchanged during the transmission from intermediate node B 22 to server node S 18. In one embodiment, if the extracted data do not match the plaintext data, server node S 18 can ignore the request $RQST_c$ 17. In other embodiments, the server node S 18 can process valid portions of $RQST_c$.

If RQST_b is valid, the above-described process is repeated by server node S 18 with respect to RQST_a 15.

Return Protocol

Fig. 6 shows general operation of the client server system 2 in the return direction from server node S 18 to client node A 14 through nodes C and B, 26 and 22, respectively. Generally, server node S 18 produces a full response, e.g., RPNSE_c 19, to a valid final request, e.g., RQST_c 17 described previously. The full response includes a response for each node in the chain of nodes requesting a session key. Each response contains the session key or keys intended for that node. Each response is also embedded within a response to a previous node as described below. As shown, RPNSE_c 19 includes RPNSE_b 20, and RPNSE_b 20 includes RPNSE_a 21.

These responses traverse the chain of nodes in the reverse order by which the final request RQST_c 17 reached server node S 18. For example, intermediate node C 26 receives a response to its request for a session key before intermediate node B 22 receives its response to its request for a session key. As such, intermediate node C 26 removes that portion of the response RPNSE_c 19 that is intended for intermediate node C 26 and forwards the remainder of the response, RPNSE_b 20, to intermediate node B 22. Likewise, intermediate node B 22 extracts that portion of the response intended for intermediate node B 22 and forwards the remainder of the response, RPNSE_a 21, to client node A 14. Thus, by this single traversal of the chain of nodes, server node S 18 can securely deliver at least one session key to each node in the chain.

Generation of the Response

Figs. 7A and 7B shows an exemplary process by which server node S 18 produces the complete response RPNSE_c 19. Server node S 18 generates RPNSE_c 19 using session keys intended for the respective nodes such that each node in the chain can access only those session keys intended for that node.

Generating RPNSE_a

Because of the recursive nesting of responses, the response intended for client node A 14, RPNSE_a 21, is the most deeply nested response within the full response, RPNSE_c 19. Accordingly, server node S 18 can construct RPNSE_c 19 starting with RPNSE_a 21. In step 120, server node S 18 generates plaintext intended for client node A 14. The plaintext can include a response message *Rab* and one or more nonces. *Rab* includes the identities of the two nodes for which the session key is intended, here nodes A and B. An exemplary notation for *Rab* is (A, B). In one embodiment, the plaintext includes nonces *Na* and *Nb*, which are the same nonces as *Na* and *Nb* sent to server node S 18 in the request RQST_c 17. Including the nonce *Na* in RPNSE_a 21 enables client node A 14 to determine that RPNSE_a 21 is current. Client node A 14 can determine from the nonce *Na* that the RPNSE_a 21 is a response from server node S 18 to a current outstanding request and not a unauthorized replication of a valid, previous communication between nodes A and S. In one embodiment, server node S 18 concatenates the nonces to the end of the response message *Rab*, which can be expressed as:

(*Rab*, *Na*, *Nb*).

The server node S 18 generates (step 124) a sealed version of the plaintext data (*Rab*, *Na*, *Nb*). In one embodiment, the sealed version includes a message digest *Dra* produced by

computing a one-way hash function of the plaintext data Rab , Na , Nb using the key Ka . An exemplary notation for the message digest Dra is

$$\langle Rab, Na, Nb \rangle Ka.$$

By including more than one nonce in the response message digest Dra , the digest is ensured to be distinct from any message digest Da created in the forward direction by client node A 14. Thus, any intruding process that captures and retains a message digest Da in the forward direction cannot use that message digest to extract the session key from the response message digest Dra .

Server node S 18 generates (step 128) a session key, Kab , which could be used by client node A 14, for example, to preserve data integrity or with encryption to preserve data confidentiality when communicating with intermediate node B 22. Server node S 18 next encodes (step 132) the session key Kab to produce encoded session key $enKab$. In one embodiment, the encoded session key Kab is generated by a bit-wise exclusive-OR (XOR) of the two bit strings. That is, the session key Kab is XOR'd with the message digest Dra :

$$enKab = (Kab \text{ XOR } \langle Rab, Na, Nb \rangle Ka).$$

In step 136, server node S 18 combines $enKab$ with the plaintext response message Rab . In one embodiment, this is accomplished by appending $enKab$ to the end of Ra . The resulting plaintext is

$$(Rab, Na, Nb, enKab),$$

which expands to

$$(Rab, Na, Nb, (Kab \text{ XOR } \langle Rab, Na, Nb \rangle Ka)).$$

Because client node A 14 is the last node in the chain of nodes to receive a response from server node S 18, the $RPNSEa$ 21 does not contain a response to a subsequent node.

Accordingly, the next step in the process of generating RPNSE_a 21 is to generate (step 144) a sealed version of the plaintext data $Rab, Na, Nb, enKab$. In one embodiment, computing a one-way hash function of the plaintext data produced in step 136, namely $Rab, Na, Nb, enKab$, using the session key Kab , produces the sealed version. As before, other sealing methods can be used.

Server node S 18 completes the response directed to client node A 14 by combining (step 148) the sealed version produced in step 144 with the plaintext produced in step 136. The resulting response to client node A 14, RPNSE_a 21, can be expressed as:

$$(Rab, Na, Nb, enKab), \langle Rab, Na, Nb, enKab \rangle Kab$$

Generating RPNSE_b

Server node S 18 generates (step 120) plaintext intended for intermediate node B 22. The plaintext includes a response message Rba and one or more nonces. Like RPNSE_a 21, the nonces in RPNSE_b 20 can be Na and Nb . Rba includes the identities of the two nodes for which an enclosed session key can be used for conducting secure communications. An exemplary notation for Rba is (B, A) . In one embodiment, server node S 18 concatenates the nonces to the end of the response message Rba , which can be expressed as:

$$(Rba, Nb, Na).$$

In step 124, server node S 18 generates a sealed version of the plaintext data Rba, Nb, Na . In one embodiment, the sealed version includes a response message digest $Drba$. Server node S 18 produces $Drba$ by computing a one-way hash function of the data Rba, Nb, Na using the key Kb shared exclusively with intermediate node B 22. An exemplary notation for the response message digest, $Drba$, is

$$\langle Rba, Nb, Na \rangle Kb.$$

Server node S 18 generates (step 128) a session key, K_{ba} , which intermediate node B 22 can use, for example, with encryption to encode communications with client node A 14. In one embodiment, the session key K_{ba} has the same value as the session key K_{ab} . Accordingly, intermediate node B 22 can use K_{ba} to decode communications encoded by client node A 14 using K_{ab} , and client node A 14 can use K_{ab} to decode communications encoded by intermediate node B 22 using K_{ba} . Server node S 18 next encodes (step 132) the session key K_{ba} to form encoded session key enK_{ba} . In one embodiment, the encoded session key K_{ba} is generated by exclusive-ORing the session key K_{ba} with the response message digest Dr_{ba} , i.e.,

$$enK_{ba} = (K_{ba} \text{ XOR } \langle R_{ba}, N_b, N_a \rangle K_b).$$

In step 136, server node S 18 combines enK_{ba} with the plaintext response message R_{ba} , for example, by appending enK_{ba} to R_{ba} . The resulting plaintext can be expressed as:

$$(R_{ba}, N_b, N_a, enK_{ba}),$$

which expands to

$$(R_{ba}, N_b, N_a, (K_{ba} \text{ XOR } \langle R_{ba}, N_b, N_a \rangle K_b)).$$

Because intermediate node B 22 is an intermediate node in the chain of nodes, the response directed to intermediate node B 22 includes the response to a subsequent node, here client node A 14. Server node S 18 combines (step 140) the response to client node A 14, $RPNSE_a$ 21, with the plaintext produced in step 136. The plaintext result of step 140 can be expressed as:

$$(R_{ba}, N_b, N_a, enK_{ba}, RPNSE_a).$$

A sealed version of this plaintext is generated (step 144), for example, by computing a one-way hash function of the plaintext data components produced in step 140 using the

session key K_{ba} . The sealed version can be expressed as $\langle R_{ba}, N_b, Na, \text{en}K_{ba}, \text{RPNSEa} \rangle K_{ba}$. Server node S 18 combines (step 148) the sealed version produced in step 144 with the plaintext produced in step 140. The result can be expressed as:

$$(R_{ba}, N_b, Na, \text{en}K_{ba}, \text{RPNSEa}), \langle R_{ba}, N_b, Na, \text{en}K_{ba}, \text{RPNSEa} \rangle K_{ba}.$$

Being an intermediate node, intermediate node B 22 communicates with neighboring nodes A and C, 14 and 26, respectfully. Intermediate node B 22, therefore, requires a session key to communicate securely with client node A 14 and another session key to communicate securely with intermediate node C 26. As described above, the session key for communicating with client node A 14 is K_{ba} . By repeating steps 120 through 136, server node S 18 includes (step 152) a second session key, K_{bc} , in the response so that intermediate node B 22 can securely communicate with intermediate node C 26. The result of repeating steps 120 through 136 with respect to K_{bc} is plaintext that can be expressed as:

$$(R_{bc}, N_b, N_c, (K_{bc} \text{ XOR } \langle R_{bc}, N_b, N_c \rangle K_b)),$$

where $(K_{bc} \text{ XOR } \langle R_{bc}, N_b, N_c \rangle K_b)$ is the encoded session key K_{bc} (hereafter $\text{en}K_{bc}$); where $\langle R_{bc}, N_b, N_c \rangle K_b$ is the message digest D_{rbc} of plaintext data R_{bc}, N_b, N_c using the shared key K_b ; where N_b and N_c are nonces that are the same as those nonces N_b and N_c received by server node S 18 in the RQSTc 17; and where R_{bc} is a response message directed to intermediate node B 22 containing the identities of the nodes, here B and C, for which the session key K_{bc} is intended to provide secure communication.

In step 156, server node S 18 combines the plaintext produced by step 152, namely $(R_{bc}, N_b, N_c, \text{en}K_{bc})$, with the result of step 148, namely $(R_{ba}, N_b, Na, \text{en}K_{ba}, \text{RPNSEa}), \langle R_{ba}, N_b, Na, \text{en}K_{ba}, \text{RPNSEa} \rangle K_{ba}$, to produce a plaintext result that can be expressed as:

$(Rbc, Nb, Nc, enKbc, Rba, Nb, Na, enKba, RPNSEa, \langle Rba, Nb, Na, enKba, RPNSEa \rangle Kba),$

In step 160, server node S 18 completes the response, $RPNSEb$ 20, by repeating steps 140 and 144 with respect to the second session key, Kbc . The resulting response $RPNSEb$ 20 can be expressed as:

$(Rbc, Nb, Nc, enKbc, Rba, Nb, Na, enKba, RPNSEa, \langle Rba, Nb, Na, enKba, RPNSEa \rangle Kba), \langle Rbc, Nb, Nc, enKbc, Rba, Nb, Na, enKba, RPNSEa, \langle Rba, Nb, Na, enKba, RPNSEa \rangle Kba \rangle Kbc.$

Generating $RPNSEc$

To generate $RPNSEc$, the process returns to step 120. Server node S 18 generates plaintext and a sealed version of the plaintext that can include a response message Rcb and one or more nonces. The plaintext can include nonces Nb and Nc , which are the same nonces, Nb and Nc , sent to server node S 18 in the request $RQSTc$ 17. Rcb includes the identities of the two nodes for which the session key is intended, here intermediate nodes B and C, 22 and 26, respectively. An exemplary notation for Rcb is (C, B) . In one embodiment, server node S 18 concatenates the nonces to the response message Rcb , which can be expressed as:

$(Rcb, Nc, Nb).$

In step 124, server node S 18 generates a sealed copy of the plaintext data components Rcb , Nc , and Nb . The sealed copy can include a message digest, Drc , produced by computing a one-way hash function of the data components Rcb , Nc , and Nb using the key Kc shared exclusively with server node S 18. An exemplary notation for the message digest Drc is

$\langle Rcb, Nc, Nb \rangle Kc$.

In step 128, server node S 18 generates a session key, Kcb , which intermediate node C 26 can use, for example, with encryption to encode communications with intermediate node B 22 so as to preserve data confidentiality. The session key Kcb has the same value as the session key Kbc . Accordingly, intermediate node C 26 can use Kcb to decode communications encoded by intermediate node B 22 using Kbc , and intermediate node B 22 can use Kbc to decode communications encoded by intermediate node C 26 using Kcb .

Server node S 18 encodes the session key Kcb to produce an encoded session key $enKcb$ (step 132). In one embodiment, $enKcb$ is generated by exclusive-ORing (XOR) the session key Kcb with the message digest Drc , i.e.,

$$enKcb = (Kcb \text{ XOR } \langle Rcb, Nc, Nb \rangle Kc).$$

In step 136, server node S 18 combines $enKcb$ with the plaintext response message Rcb , for example, by appending $enKcb$ to Rcb to produce the following plaintext:

$$(Rcb, Nc, Nb, enKcb).$$

Because intermediate node C 26 is an intermediate node in the chain of nodes, the response directed to intermediate node C 26 includes the response to a subsequent node, here intermediate node B 22. Server node S 18 combines (step 140) the response to intermediate node B 22, $RPNSEb$ 20, with the plaintext produced in step 136. The plaintext result of step 140 can be expressed as:

$$(Rcb, Nc, Nb, enKcb, RPNSEb).$$

In step 144, a sealed version of this plaintext is generated, for example, by computing a one-way hash function of the plaintext data produced in step 140, namely Rcb , Nc , Nb , $enKcb$, and $RPNSEb$, using the session key Kcb . The resulting sealed version is:

$\langle R_{cb}, N_c, N_b, \text{en}K_{cb}, \text{RPNSE}_b \rangle K_{cb}$.

In one embodiment, the last node in the chain before server node S 18 requires only one session key. Accordingly, in step 148, server node S 18 can complete the response RPNSE_c 19 by combining the sealed version produced in step 144 with the plaintext produced in step 136.

Alternatively, intermediate node C 26, as an intermediate node, can require another session key for communicating with server node S 18. In such an embodiment, the generation of RPNSE_c 19 continues by encapsulating a second session key in a similar manner as used to include a second session key in RPNSE_b 20 described above.

When intermediate node C 26 does not require two session keys, the final response, RPNSE_c , passing from server node S 18 to intermediate node C 26 can be expressed as:

$(R_{cb}, N_c, N_b, \text{en}K_{cb}, \text{RPNSE}_b), \langle R_{cb}, N_c, N_b, \text{en}K_{cb}, \text{RPNSE}_b \rangle K_{cb}$.

The response to each node X requesting a session key generalizes to:

$\text{RPNSE}_x = (... , \text{en}K_{x,x+1}, (... , \text{en}K_{x,x-1}, \text{RPNSE}_{x-1}), \langle ... , \text{en}K_{x,x-1}, \text{RPNSE}_{x-1} \rangle K_{x,x-1}),$
 $\langle ... , \text{en}K_{x,x+1}, (... , \text{en}K_{x,x-1}, ... \text{RPNSE}_{x-1}), \langle ... , \text{en}K_{x,x-1}, \text{RPNSE}_{x-1} \rangle K_{x,x-1}$
 $1 \rangle K_{x,x+1},$

where X is the identity of the node targeted for the RPNSE_x , X+1 is the identity of the neighboring node toward the server 18, and X-1 is the identity of the neighboring node away from the server 18. Note that the generalized form applies to intermediate nodes requiring two session keys, and that in some implementations the first and last nodes in the chain (e.g., client node A 14 and intermediate node C 26) may receive only one session key.

Protocol for Processing Server Response

Fig. 8 shows an exemplary process by a full response from server node S 18 is processed by each node in a chain of nodes to obtain individual session keys.

Evaluating RPNSE_c

For illustration purposes, intermediate node C 26 is the first node in the chain of nodes to receive the response, here RPNSE_c 19, which has been generated and issued by server node S 18 (step 164). As described above, RPNSE_c 19 includes a plaintext portion and a sealed portion. The plaintext portion includes R_{cb} , N_c , N_b , enK_{cb} , and the next embedded response RPNSE_b 20. Intermediate node C 26 cannot decipher any of the session keys sealed within RPNSE_b 20 because RPNSE_b is sealed with key K_b .

Because intermediate node C 26 has the key K_c , intermediate node C 26 can generate (step 168) the message digest D_{rc} from the plaintext components R_{cb} , N_c , N_b included in the response RPNSE_c 19. Intermediate node C 26 then uses the message digest D_{rc} to extract the session key K_{cb} from the plaintext by performing a bit-wise exclusive-OR operation using enK_{cb} and the message digest D_{rc} (step 172). This exclusive-OR operation recovers K_{cb} because exclusive ORing twice in succession with the same value reproduces the original value. For example:

- 1). $K_{cb} \text{ XOR } D_{rc} = enK_{cb};$
- 2). $enK_{cb} \text{ XOR } D_{rc} = K_{cb}.$

Once intermediate node C 26 obtains K_{cb} , intermediate node C 26 uses (step 176) K_{cb} to extract the data within the sealed portion of RPNSE_c 19. The extracted data include the embedded response RPNSE_b 20. Intermediate node C 26 then compares (step 180) the extracted data with the plaintext data.

If the extracted data do not match the plaintext data, RPNSE_c 19 is invalid. If instead the extracted data match the plaintext data, intermediate node C 26 knows that server node S 18 is the sender of the RPNSE_c 19 because only server node S 18 and intermediate node C 26

know the key K_c used to produce the encoded session key enK_{cb} . Intermediate node C 26 also knows that the data R_{cb} , N_c , N_b , and $RPNSE_b$ are unaltered during the transmission to intermediate node C 26 from server node S 18. Further, intermediate node C 26 can compare the nonce N_c received in $RPNSE_c$ 19 with the nonce that intermediate node C 26 stored upon sending $RQST_c$ 17 to server node S 18. A match indicates the $RPNSE_c$ 19 is a current response to an outstanding request. Intermediate node C 26 can now use the extracted session key K_{cb} to encode subsequent communications with intermediate node B 22. If $RPNSE_c$ 19 is valid, intermediate node C 26 transmits (step 184) the embedded response, $RPNSE_b$ 20, to intermediate node B 22.

Evaluating $RPNSE_b$

Intermediate node B 22 receives (step 164) $RPNSE_b$ 20, which also includes plaintext and sealed data. As described above, plaintext portion of $RPNSE_b$ 20 is

$(R_{bc}, N_b, N_c, enK_{bc}, R_{ba}, N_b, N_a, enK_{ba}, RPNSE_a), \langle R_{ba}, N_b, N_a, enK_{ba}, RPNSE_a \rangle_{K_{ba}}$.

Because intermediate node B 22 has the key K_b , intermediate node B 22 can generate (step 168) the message digests Dr_{bc} and Dr_{ba} from the plaintext components included in the response $RPNSE_b$ 20. Intermediate node B 22 uses (step 172) the message digest Dr_{bc} to extract the session key K_{bc} by exclusive-ORing enK_{bc} with the message digest Dr_{bc} . Likewise, intermediate node B 22 uses the message digest Dr_{ba} to extract the session key K_{ba} by exclusive-ORing enK_{ba} with the message digest Dr_{ba} . Upon extracting (step 176) K_{bc} , intermediate node B 22 uses K_{bc} to extract the data within the sealed version of $RPNSE_b$ 20. A portion of these extracted data requires further extraction because the session key K_{ba} also seals that portion. Intermediate node B 22 then uses K_{ba} to extract the data

within this portion, one of such data being the embedded response $RPNSEa$. Intermediate node B 22 then compares (step 180) the extracted data with the plaintext data.

Again, if the extracted data do not match the plaintext data, $RPNSEb$ 20 is invalid. If instead the extracted data match the plaintext data, intermediate node B 22 knows that server node S 18 is the generator of the $RPNSEb$ 20 because only intermediate node B 22 and server node S 18 has the key Kb used to create the encoded session keys $enKbc$ and $enKba$. Because of the match, intermediate node B 22 also knows that the data Rbc , Rba , Nc , Nb , Na , and $RPNSEa$ are unaltered during the transmission to intermediate node B 22 from server node S 18. Further, intermediate node B 22 can compare the nonce Nb received in $RPNSEb$ 20 with the nonce that intermediate node B 22 stored upon sending $RQSTb$ 16 to intermediate node C 26. A match indicates the $RPNSEb$ 20 is a current response to an outstanding request. Intermediate node B 22 can now use the extracted session key Kba to encode subsequent communications with client node A 14 and the extracted session key Kbc to encode subsequent communications with intermediate node C 26. If $RPNSEb$ 20 is valid, intermediate node B 22 transmits (step 184) the embedded response, $RPNSEa$ 21, to client node A 14.

Evaluating $RPNSEa$

Upon receiving $RPNSEa$ 21, client node A 14 evaluates $RPNSEa$ 21 in a manner similar to the evaluation of $RPNSEc$ 19 by intermediate node C 26. Client node A 14 receives (step 164) $RPNSEa$ 21 from intermediate node B 22. $RPNSEa$ 21 includes plaintext and sealed data. As described above, the plaintext portion of $RPNSEa$ 21 is

$(Rab, Na, Nb, enKab)$.

Because client node A 14 has the key K_a , client node A 14 can generate (step 168) the message digest D_{ra} from the plaintext components R_{ab} , N_a , N_b included in the response $RPNSE_a$ 21. Client node A 14 then uses (step 172) the message digest D_{ra} to extract the session key K_{ab} from the plaintext by exclusive-ORing enK_{ab} with the message digest D_{ra} . After obtaining the session key K_{ab} , client node A 14 uses (step 176) K_{ab} to extract the data within the sealed portion of $RPNSE_a$ 21. Client node A 14 then compares (step 180) the extracted data with the plaintext data. Again, if the extracted data do not match the plaintext data, client node A 14 can ignore $RPNSE_a$ 21 because $RPNSE_a$ 21 is invalid. If the extracted data match the plaintext data, client node A 14 knows that server node S 18 is the generator of the $RPNSE_a$ 21 because only server node S 18 and client node A 14 have the key K_a used to create the encoded session key enK_{ab} . Client node A 14 also knows that the data R_{ab} , N_a , N_b , and $RPNSE_a$ are unaltered during the transmission to client node A 14 from server node S 18.

Further, client node A 14 can compare the nonce N_a received in $RPNSE_a$ 21 with the nonce that client node A 14 stored upon sending $RQST_a$ 15 to intermediate node B 22. A match indicates the $RPNSE_a$ 21 is a current response to an outstanding request. Accordingly, client node A 14 can use the extracted session key K_{ab} to encode subsequent communications with intermediate node B 22. At client node A 14, the propagation of responses completes. With one round-trip traversal of the forward and reverse paths, each node in a chain of nodes has securely requested and securely received at least one session key from the server node S 18.

While the invention has been shown and described with reference to specific preferred embodiments, it should be understood by those skilled in the art that various

changes in form and detail may be made without departing from the inventive concepts disclosed.

CLAIMS

1. In a network including a first node, a second node, and a third node, a method for securely delivering digital information to the first node from the third node by way of the second node, the method comprising the steps of:
 - receiving a request at the third node from the first node;
 - generating digital information in response to the request;
 - operating on the request and the digital information to produce a first data structure, the first data structure including a representation of the digital information;
 - operating on the request and the first data structure to produce a second data structure, the second data structure including the first data structure; and
 - transmitting the second data structure to the second node.
2. The method of claim 1 wherein the digital information is a session key for the first node to use in communications with the second node.
3. The method of claim 2 further comprising the step of encoding the session key using a key shared exclusively with the first node to conceal the session key within the first data structure.
4. The method of claim 2 further comprising the step of sealing a portion of the first data structure using the session key.
5. The method of claim 4 further comprising the steps of:
 - receiving the first data structure from the second node;
 - decoding the encoded session key using the shared key;

extracting data from the sealed portion of the first data structure using the decoded session key; and

using the extracted data (1) to authenticate that the session key originated from the third node, (2) to determine that the session key is unaltered during transmission from the third node, and (3) to determine that the first data structure is a current response from the third node to the request from the first node.

6. The method of claim 2 further comprising the steps of:

generating a second session key for the second node to use in communications with the first node;

sealing a portion of the second data structure containing the first data structure using the second session key.

7. The method of claim 6 further comprising the steps of:

encoding the second session key using a key shared exclusively with the second node; and

including the encoded second session key within the second data structure.

8. The method of claim 2 further comprising the steps of:

extracting the first data structure from the second data structure at the second node; transmitting the first data structure to the first node from the second node; and extracting the session key from the first data structure at the first node.

9. The method of claim 8 wherein the session key is a first session key and the second data structure includes a second session key and further comprising the step of:

extracting the second session key from the second data structure at the second node, and wherein the first and second session keys provide secure communication between the first node and the second node.

10. The method of claim 1 wherein the second node is a first intermediate node and the network includes a second intermediate node in a communication path between the first intermediate node and the third node, and further comprising the steps:

operating on the request and the second data structure to generate a third data structure, the third data structure including the second data structure;

transmitting the third data structure to the second intermediate node; and

extracting the second data structure at the second intermediate node for transmission to the first intermediate node.

11. The method of claim 2 wherein the step of generating the first data structure includes:

generating plaintext;

encoding the session key;

generating a digest of a combination of the plaintext and the encoded session key; and

combining the plaintext, the encoded session key, and the digest to produce the first data structure.

12. The method of claim 11 wherein the step of generating the digest includes applying a one-way hash function using the session key to the combination of the plaintext and the encoded session key.

13. The method of claim 11 wherein the step of generating the digest includes applying an encryption algorithm using the session key to the combination of the plaintext message and the encoded session key.

14. The method of claim 11 wherein the step of encoding the session key includes:
generating a digest of the plaintext; and
exclusive-ORing the session key with the digest of the plaintext to produce the encoded session key.

15. The method of claim 11 wherein the plaintext includes a first nonce associated with the first node and a second nonce associated with the second node.

16. The method of claim 2 wherein the step of generating the second data structure includes:

generating plaintext;
generating a second session key;
encoding the second session key;
generating a digest of a combination of the plaintext, the encoded second session key, and the first data structure; and
combining the plaintext, the encoded second session key, the first data structure, and the digest to produce the second data structure.

17. The method of claim 2 further comprising the step of generating the request including:

generating a first plaintext at the first node;

generating a first digest of the first plaintext at the first node;
transmitting a first combination of the first plaintext and the first digest from the first node to the second node;
generating a second plaintext at the second node;
generating a second digest of a second combination of the second plaintext and the first combination of the first plaintext and the first digest; and
combining the second plaintext and the second digest to produce the request.

18. In a network including a client node and an authentication server node, a method for securely delivering a session key to the client node from the authentication server node in response to a request from the client node, the method comprising the steps of:

sealing plaintext using the session key;
encoding the session key using a key shared with the client node; and
transmitting a data structure to the client node that includes the encoded session key and the sealed plaintext.

19. The method of claim 18 further comprising the steps of:

receiving the data structure;
decoding the encoded session key using the shared key;
extracting the sealed plaintext using the decoded session key; and
using the extracted plaintext to authenticate that the session key originated from the authentication server.

20. The method of claim 19, further comprising the step of determining from the extracted plaintext that the decoded session key is unaltered during transmission from the authentication server.

21. The method claim 19, further comprising the step of determining from the extracted plaintext that the data structure is a current response from the authentication server to the request from the client node.

22. The method of claim 18 wherein the data structure is a first data structure and the network includes an intermediate node in a communication path between the authentication server node and the client node, and further comprising the steps:

operating on the request and the first data structure to generate a second data structure, the second data structure including the first data structure; and

transmitting the second data structure to the intermediate node for extracting the first data structure at the intermediate node and for transmitting the extracted first data structure to the client node.

23. A system for securely distributing a session key by way of a network, the network including a first node transmitting a request to obtain the session key and a second node in communication with the first node, the system comprising:

a third node in communication with the second node and receiving the request by way of the second node, the third node including:

a processor generating (1) a first data structure by operating on the request and the session key, the first data structure including a representation of the session key, and (2)

generating a second data structure by operating on the request and the first data structure, the second data structure including the first data structure; and

a network interface coupled to the processor for transmitting the second data structure to the second node over the network.

24. A method for securely delivering digital information in a network, substantially as hereinbefore described with reference to the accompanying drawings.

25. A method for securely delivering a session key to a client node from an authentication server node in a network, substantially as hereinbefore described with reference to the accompanying drawings.

26. A system for securely distributing a session key by way of a network, substantially as hereinbefore described with reference to the accompanying drawings.



Application No: GB 9925678.6
Claims searched: 1-17,23,24,26

Examiner: Owen Wheeler
Date of search: 5 May 2000

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:
UK CI (Ed.R): H4P (PDCSP, PPEB)
Int CI (Ed.7): H04L: 9/08, 12/22
Other: Online: EPODOC, JAPIO, WPI

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
X	EP 0851628 A1	[ICO] See Figs. 1,10-13 and 17. Column 11 line 29 to column 13 line 46 and column 14 line 32 to column 15 line 9.	1,2,3,23
X	US 5491750 A	[BELLARE] See Figs. 4-6 and column 10 line 18 to column 12 line 4.	1-3,8,9, 11-16,23

40

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.